# Atac al problema del logaritme discret mitjançant la paral·lelització de l' algorisme Rho de Pollard emprant la plataforma CoDiP2P

Universitat de Lleida
Escola Politècnica Superior
Enginyeria Tècnica en Informàtica de Sistemes

Treball Final de Carrera

Setembre de 2011

Autor: Adrià Díez Casamiquela
Codirectors: Fernando Cores Prado
Ramiro Moreno Chiral

Vull agraïr als meus dos tutors de treball,
al Ramiro, pels coneixements tant acadèmics
com personals que m' ha transferit, com al Fernando,
pel seu incansable esforç i gran paciència,
a la meva família i amics.

# Índex

# Índex de taules

# Índex de figures

# Llista d'algorismes

# Capítol 1

## 1 Pròleg

El sorgiment i posterior desenvolupament de les xarxes de comunicació, en particular Internet, ha obert en les darreres dues dècades una nova possibilitat de comunicació, és a dir, d' intercanvi d' informació en la societat. Aquest fet ha provocat de manera imparable que Internet esdevingués font de recursos i serveis i, per tant, es convertís també en plataforma sobre la que funcionessin negocis diversos. Alhora les amenaces a la seguretat de la informació que es transmet a través d' Internet han crescut progressivament. És necessari, doncs, crear diferents mecanismes dirigits a garantir la confidencialitat i autenticitat dels documents electrònics. Aquest conjunt de mecanismes conformen l' anomenada *seguretat informàtica*. La *seguretat informàtica* consisteix en un compendi d' especialitats, tècniques, disciplines, protocols, etc. que funcionen des del més bàsic de la seguretat física dels sistemes informàtics fins a les tècniques de xifrat i protocols criptogràfics més complexos, destacant també les polítiques de seguretat, recuperació de dades, auditories, etc.

L' estudi de qualsevol algorisme criptogràfic en informàtica va lligat a la capacitat de còmput dels computadors actuals. Degut a l' impressionant increment de potencial dels ordinadors d' avui en dia es posa de manifest la necessitat de nous sistemes de xifrat basats en problemes matemàtics inabastables que permetin garantir la seguretat de la informació que es vol transmetre en una xarxa com és Internet, cada cop més expansiva. En contraposició a la ciència de la criptografia, el criptoanàlisi busca desxifrar els missatges codificats. Així doncs, mentre la *criptografia* es dedica a desenvolupar la codificació de la informació (missatges), el *criptoanàlisi* es dedica al contrari, a desenvolupar tècniques per a desxifrar o descodificar la informació protegida amb aquests recursos.

Aquest treball consisteix precisament en el criptoanàlisi del problema del *logaritme discret* que s' exposarà en els propers capítols. Per a fer-ho, s' empra l' algorisme *Rho de Pollard* sobre la plataforma *CoDiP2P*, un sistema de computació distribuida peer to peer.

Aquest treball té com a objectiu l' estudi dels següents conceptes:

- El problema del logaritme discret (DLP).
- L' algorisme Rho de Pollard i les seves paral·lelitzacions.
- La plataforma CoDiP2P.

# Capítol 2

## 2 Introducció

Durant milers d' anys, les grans civilitzacions han depès de la comunicació eficient per a dominar els seus territoris. Alhora, tots han estat conscients de les conseqüències que es produïen si els seus missatges eren interceptats pels estats oponents. Per aquest motiu, es van desenvolupar els codis i les xifres, tècniques per a amagar (*kryptós*, en grec) un missatge o escrit (*gráphin*, que vol dir escriure) de tal manera que només les persones autoritzades poguessin tenir accés al missatge. Naixia la criptografia. Paral·lelament a l' esteganografia, que consistia en l' ocultació del missatge, es va produir l' evolució de la criptografia, que tractava d' ocultar el significat del missatge, procés conegut amb el terme de *codificació*. L' evolució de la història va lligada a l' evolució de la criptografia i, per tant, de les xifres.

Des del primer mètode criptogràfic, l' escítala, els sistemes de xifrat van evolucionar millorant la seva robustesa enfront el criptoanàlisi (del grec *kryptós*, "amagat" i *analýein*, "desfer"), estudi dels mètodes per a obtenir el sentit de la informació xifrada, sense accedir a la informació secreta necessària per a obtenir el sentit del missatge. A continuació es citen alguns dels mètodes més emprats al llarg de la història: els mètodes de *substitució* (el més conegut, anomenat xifrat de Cèsar) i transposició *monoalfabètics*, anomenats criptosistemes clàssics, els mètodes o *sistemes de substitució polialfabètica*, impulsats per León Battista Alberti i desenvolupats posteriorment per Blaise de Vigenère, Giovanni Porta i Johannes Trithemius, *la màquina Enigma*, fabricada per Arthur Scherbius, etc.

Els avenços en computació automàtica suposen una amenaça evident tant als sistemes de xifrat existents com una oportunitat per al desenvolupament de nous sistemes criptogràfics. A mitjan anys 70 del segle XX, l' autoritat nordamericana d' estàndards NBS (National Bureau of Standards) publica el primer disseny del sistema criptogràfic *DES*. Anys més tard, el succeeixen els algoritmes 3DES i AES. Aquest és l' inici de la criptografia moderna i l' etapa presuccessora dels sistemes asimètrics. Aquests sistemes suposaren un salt qualitatiu ja que permetien introduir la criptografia en camps tan essencials com el de la signatura digital. L' any 1976 es publica l' article de Diffie-Hellman [17], en el qual es proposava la utilització de la clau pública en els criptosistemes. Aquest canvi en la concepció dels criptosistemes va causar una autèntica revolució de la criptografia teòrica i pràctica. Els primers que van inventar un criptosistema de clau asimètrica o pública basat en una funció matemàtica d' una sola via[1]

[1] Una *funció matemàtica d' una sola via* és aquella on l' operació matemàtica directa és fàcil d' efectuar però no ho és l' operació inversa. Podem imaginar una funció d' una sola via a la unió de pintures: és senzill unir dues pintures de colors diferents però no ho és separar-les

foren Ron Rivest, Adi Shamir i Leonard Adleman amb el sistema RSA [1], l' any 1978. La seguretat d' aquest algorisme rau en el problema de la factorització de nombres enters[2], una funció matemàtica d' una sola via ja que la generació de qualsevol nombre enter és computacionalment senzill (producte dels seus divisors excepte ells mateixos), però no ho és així la factorització d' aquest nombre. Així com el problema de la factorització de nombres enters, el problema del logaritme discret (DLP) esdevé un altre problema en què es basen bona part dels criptosistemes de clau pública.

Aquest treball de final de carrera té com a objectiu l' estudi del problema del logaritme discret i les variants d' un dels possibles i millors atacs que existeixen fins ara, la Rho de Pollard. En primer terme, s' exposen els conceptes matemàtics i criptogràfics necessaris per tal d' entendre correctament els capítols que es succeeixen. A continuació s' explica en què consisteix el problema del logaritme discret i quins possibles atacs presenta. Seguidament, es tracta la implementació de les variants utilitzades de la Rho de Pollard sobre el grup multiplicatiu $\mathbb{F}_p^*$, paral·lelitzant l' algorisme sobre la plataforma CoDiP2P. En darrer terme, es realitza una anàlisi complerta dels resultats obtinguts en les diverses proves realitzades i un conjunt de conclusions a què s' ha arribat sobre el treball realitzat.

un cop unides. En termes criptogràfics, una funció d' una sola via és aquella on no existeix l' operació de desxifrar, és a dir, a partir d' un missatge xifrat no és possible obtenir el missatge en clar que l' ha generat.

[2]Es tracta d' una consideració general però que encara no s' ha comprovat.

# Capítol 3

# 3 Nocions prèvies

En aquest capítol s' exposen els fonaments matemàtics, criptogràfics i relacionats amb CoDiP2P necessaris per a la correcta comprensió del treball.

## 3.1 Fonaments Matemàtics

A continuació s' exposen conceptes directament relacionats amb la matemàtica emprada en aquest treball, conceptes de quatre branques de la matemàtica: les teories de conjunts, grups, anells i cossos.

### 3.1.1 Teoria de Conjunts

**Definició 3.1.1.1**

Un *conjunt* és una agrupació o col·lecció, concebuda com una entitat d' objectes ben diferenciats i definits. Els objectes que formen un conjunt s' anomenen *elements* d' aquest conjunt.

**Definició 3.1.1.2**

El *cardinal* d' un conjunt és el nombre d' elements que té aquest conjunt. Es denota per *Card(G)* , $|G|$ o $\#G$.

### 3.1.2 Teoria de Grups

**Definició 3.1.2.1**

Un conjunt $G$ dotat d' una operació interna[3] $*$ té estructura algebraica de *grup* o $(G, *)$ és un grup si se satisfan les següents propietats:

1. Associativa: $a * (b * c) = (a * b) * c$, $\forall a, b, c \in G$;
2. Existeix un únic *element* neutre $e \in G : a*e = e*a$, $\forall a \in G$;
3. Tot element és invertible o simetritzable, és a dir, existeix un únic element $a^{-1} \in G$, anomenat *simètric* o *invers* de $a$, tal que $a*a^{-1} = a^{-1}*a = e$, on $e$ és l' element neutre de $(G, *)$.

Si, a més, se satisfà la propietat commutativa, és a dir $a*b = b*a$, $\forall a, b \in G$, es diu que $(G, *)$ és un *grup abelià*.

[3] Una *llei de composició interna* o *operació interna* $*$ sobre un conjunt G és aquella que satisfà: $a*b \in G$, $\forall a, b \in G$.

**Definició 3.1.2.2**

Un grup $(G, *)$ és *finit* si *Card(G )* és finit.

**Definició 3.1.2.3**

Un grup $(G, *)$ és *cíclic* si $\exists g \in G : G = \{g^n : n \in \mathbb{Z}\}$, on $g$ és generador del grup $G$ . És a dir, qualsevol element de $G$ pot expressar-se com una potència de $g$.

**Definició 3.1.2.4**

L' ordre d' un grup $(G, *)$ és el cardinal del conjunt $G$.

**Definició 3.1.2.5**

L' ordre d' un element $a \in G$ és $(G, *)$ és el menor

$n \in \mathbb{N} : a^n = a * \overset{n)}{\cdots} * a = e$, on $e$ és l' element neutre de $(G, *)$.

**Definició 3.1.2.6**

Siguin$(G, *)$ i $(G', \cdot)$ dos grups amb les seves respectives operacions, s' anomena *morfisme de grups* a qualsevol aplicació

$$\begin{aligned} f : G &\longrightarrow G' \\ a &\longmapsto f(a), \end{aligned}$$

que satisfà $f(a*b) = f(a) \cdot f(b)$, $\forall a, b \in G$.

### 3.1.3 Teoria d' Anells

**Definició 3.1.3.1**

Un conjunt $G$ dotat de dues operacions internes $+$ i $\cdot$ té estructura d' *anell* o $(G, +, \cdot)$ és anell si se satisfan les següents propietats:

1. $(G, +)$ és un grup abelià;
2. L' operació $\cdot$ és associativa;
3. L' operació $\cdot$ és distributiva respecte de l' operació $+$.[4]

[4]Donades dues operacions internes $+$ i $\cdot$ sobre un conjunt $G$ , l' operació $\cdot$ és distributiva respecte de l' operació $+$, si se satisfà: $a \cdot (b + c) = a \cdot b + a \cdot c$ i $(a + b) \cdot c = a \cdot c + b \cdot c$, $\forall a, b, c \in G$.

**Definició 3.1.3.2**

Un *divisor de zero* d' un anell $(G,+,\cdot)$ és un element que tot i ser diferent de zero al multiplicar-lo per un altre element també diferent de zero pot donar zero. És a dir, $a \in G-\{0\}$ és un divisor de zero si $\exists b \in G-\{0\} : a\cdot b = 0$ o bé $b\cdot a = 0$, on 0 és l' element neutre de $(G,+)$. L' anell $(G,+,\cdot)$ és *unitari* si té element neutre respecte de l' operació $\cdot$ . L' anell $(G,+,\cdot)$ és *commutatiu* si l' operació $\cdot$ satisfà la propietat commutativa.

**L' anell de residus $\mathbb{Z}/m\mathbb{Z}$**

El conjunt quocient de $\mathbb{Z}$ per la relació de congruència mòdul $m$ , que es representa per $\mathbb{Z}_m$, $\mathbb{Z}/(m)$ o $\mathbb{Z}/m\mathbb{Z}$, està format per totes les classes de congruència diferents mòdul $m$ , és a dir,

$$\mathbb{Z}_m = \left\{\overline{0},\overline{1},\ldots,\overline{m-1}\right\}, on\, \overline{a} = \{a+k\cdot m : k\in\mathbb{Z}\}.$$

$(\mathbb{Z}/m\mathbb{Z})^*$ és el conjunt d' element invertibles de $\mathbb{Z}/m\mathbb{Z}$. Aquest conjunt amb l' operació $\cdot$ té estructura de grup abelià.

**Proposició 3.1.3.3**

Siguin $m$ un enter positiu i $a$ un enter qualsevol,

- Si $\mathrm{mcd}(a\,,m\,) = 1$, llavors $\overline{a}$ és invertible a $\mathbb{Z}/m\mathbb{Z}$, és a dir, $\exists x \in \mathbb{Z} : \overline{a}\cdot\overline{x} = \overline{1}$.

### 3.1.4 Teoria de Cossos

**Proposició 3.1.4.1**

Un conjunt $\mathbb{K}$ dotat de dues operacions internes $+$ i $\cdot$ és un *cos* si se satisfan les següents propietats:

- $(\mathbb{K},+,\cdot)$ és un anell unitari;
- Tot element de $\mathbb{K}$ diferent del 0 (element neutre de $(\mathbb{K},+)$) és invertible, és a dir,
$$\exists a^{-1} \in \mathbb{K} : a\cdot a^{-1} = a^{-1}\cdot a = 1, \forall a \in \mathbb{K}-\{0\},$$
on 1 és l' element neutre de $(\mathbb{K},\cdot)$.

  Si l' operació, a més, satisfà la propietat commutativa, es diu que $(\mathbb{K},+,\cdot)$ és un cos commutatiu.

  Pel que fa al conjunt dels nombres enters, $(\mathbb{Z}_m,+)$ és un grup abelià i $(\mathbb{Z}_m,+,\cdot)$ és un anell unitari i commutatiu que té divisors de zero si $m$ no és primer. En cas que el mòdul $m$ sigui primer, $(\mathbb{Z}_m,+,\cdot)$ té estructura de cos i es denota amb el símbol $\mathbb{F}_m$.

## 3.2 Fonaments Criptogràfics

A continuació s' exposen conceptes criptogràfics bàsics que intervenen directament en aquest projecte. Els conceptes matemàtics exposats en la secció anterior són la base de la criptografia tractada en aquest treball.

### 3.2.1 Criptologia

La *criptologia* és la ciència o estudi que té per objectiu amagar la informació continguda en certes dades, així com transformar-les per diversos motius. Esdevé l' estudi dels criptosistemes i les seves principals àrees d' estudi son la criptografia i el criptoanàlisi, tot i que també s' inclou l' esteganografia.

### 3.2.2 Criptografia

La *criptografia* és la part de la criptologia que es basa en cercar sistemes de xifrat i desxifrat de la informació mitjançant claus.

### 3.2.3 Criptoanàlisi

El *criptoanàlisi* és el conjunt de tècniques emprades per a desxifrar codis encriptats sense conèixer el sistema emprat per a la codificació. Sovint, el criptoanàlisi implica cercar la clau secreta emprada en el xifrat.

### 3.2.4 Criptograma

Un *criptograma* és un missatge xifrat el significat del qual resulta inintel·ligible fins que és desxifrat.

### 3.2.5 Criptosistema

Un *criptosistema* és una col·lecció de transformacions de texts en clar en text xifrat i viceversa, en què les transformacions que s' han d' emprar són seleccionades per claus i són definides, normalment, per un algorisme matemàtic.

#### Criptografia moderna

El camp modern de la criptografia es pot dividir en diverses àrees d' estudi: la *criptografia de clau privada* i la *criptografia de clau pública*.

#### Criptografia de clau privada

La *criptografia de clau privada*[5], també anomenada de *clau simètrica*, *secreta* o *compartida* es refereix a aquelles tècniques o mètodes de xifrat en què tant emissor com receptor comparteixen una única clau. Aquesta serveix a l' emissor per a xifrar el missatge abans d' enviar-lo i al receptor per a desxifrar-lo en rebre'l.

[5] Fou l' únic mètode de xifratge conegut públicament fins al 1976.

- Avantatges: L' emissor i receptor coneixen la clau acordada i poden enviar els seus missatges xifrats de forma segura.
- Desavantatges: Dos problemes en l' intercanvi de claus:

1. Incapacitat per trobar un canal segur per a intercanviar la clau.
2. Per a un grup d' $n$ persones, és necessari distribuir una quantitat de claus equivalent a $\frac{n \cdot (n-1)}{2}$.

Alguns exemples de criptosistemes de clau simètrica són: DES, 3DES, RC5, Blowfish, IDEA i AES, amb les seves variants CBC, OFB o CFB.

### Criptografia de clau pública

La *criptografia de clau pública,* també anomenada *asimètrica*, es refereix a aquelles tècniques o mètodes criptogràfics que empren un parell de claus, ambdues de la mateixa persona emissora, per a l' enviament de missatges.
Una d' aquestes claus és pública i pot entregar-se a qualsevol persona, l' altra clau és privada i el propietari ha de guardar-la de tal manera que ningú tingui accés a ella.
Si l' emissor empra la clau pública del receptor per a xifrar el missatge, un cop xifrat, només la clau privada del destinatari podrà desxifrar el missatge. És així com es garanteix la *confidencialitat* de l' enviament del missatge.
L' *identificació* i *autenticació* de l' emissor es garanteix xifrant el missatge, a més, amb la seva clau privada.
És d' aquesta manera com el receptor, en rebre el missatge, el desxifra amb la clau pública de l' emissor i la seva clau privada.

- Avantatges: Aquests sistemes posen fi al problema de l' intercanvi de claus dels sistemes de xifrat simètrics. Es necessiten només $n$ parell de claus per cada $n$ persones que desitgin comunicar-se entre si.
- Desavantatges:

1. Es necessita un major temps de procés per a una mateixa longitud de clau i missatge.
2. Les claus han de ser de major tamany que les simètriques.[6]
3. El missatge xifrat ocupa més espai que l' original.
4. L' atac de l' intermediari [18].

Alguns exemples de criptosistemes de clau asimètrica són: Diffie-Hellman, RSA, DSA, ElGamal i Criptografia de corba el·líptica.

[6] El *sistema de criptografia de corba el·líptica* representa una alternativa menys costosa per aquest tipus de problemes.

#### Criptografia híbrida

La *criptografia híbrida* és un mètode criptogràfic que empra tant xifrat simètric com xifrat asimètric. Empra el xifrat de clau pública per a compartir una clau per al xifrat simètric. El missatge que s' envia es xifra emprant la clau i enviant-lo al destinatari. Com que compartir una clau simètrica no és segur, la clau emprada és diferent per a cada sessió.

Tant PGP[7] com GnuPGP empren sistemes de xifrat híbrids.

## 3.3 Fonaments de CoDiP2P

*CoDiP2P* és un sistema de computació distribuida peer to peer creat a la Universitat de Lleida[8] el 2008[9]. Els propis autors defineixen el sistema amb les seguents paraules:

> "CoDiP2P[10] es tracta d' una agregació de diferents recursos per a formar un sistema col·laboratiu capaç de resoldre problemes computacionalment intractrables per a sistemes tradicionals. La potència del sistema es troba en el nombre de nodes capaços de treballar conjuntament, per aquest motiu la heterogeneïtat dels recursos té un pes molt important.[16]"

### 3.3.1 Peer-to-peer

Una *xarxa d' igual a igual*, *peer-to-peer* o *P2P* és una xarxa informàtica constituïda entre iguals. Es refereix a una xarxa que no té clients ni servidors fixos sinó una sèrie de nodes que es comporten simultàniament com a clients i com a servidors dels demés nodes de la xarxa.

### 3.3.2 Peer

Un *peer* és la unitat més petita que es pot trobar en una xarxa *p2p*. També conegut com a node, el peer s' emprarà per aprofitar els seus recursos computacionals ociosos: capacitat de còmput, d' emmagatzemament, etc. El concepte de peer engloba tant als nodes treballadors (*workers*) com als gestors (*managers*).
Un *worker* és un dels tipus de rol que pot desenvolupar un peer dins el sistema. Els workers són els encarregats d' executar les tasques que el manager que els

[7] *Pretty Good Privacy* o *PGP* és un programa desenvolupat per Phil Zimmermann la finalitat del qual és protegir la informació distribuida a través d' Internet.

[8] Els autors del treball final de màster són n' Ignasi Barri Vilardell i en Josep Maria Rius Torrentó.

[9] El primer prototip es realitza el 2006 amb el nom de CompP2P. Els autors del prototip són l'Íñigo Goiri Presa i en Josep Maria Rius Torrentó.

[10] CoDiP2P està construit a partir de la plataforma peer-to-peer de codi obert JXTA (Juxtapose) de Sun Mycrosystems.

controla els ha planificat i comunicar-los els events que puguin succeïr.

### 3.3.3 NMajor

*NMajor* és un concepte fonamental sense el qual el sistema no podria iniciar-se. Es tracta d' un servidor estàtic que té la funcionalitat de ser el punt d' entrada del sistema. La seva tasca consisteix en la cerca, registre i actualització d' una llista de nodes per a facilitar l' accés de nous usuaris al sistema.

### 3.3.4 Aplicació

L' aplicació de què es parla a l' entorn CoDiP2P és un programa implementat amb el llenguatge Java que segueix les directrius de programació paral·lelitzable, amb la finalitat que sigui fàcilment distribuïble.
Tota aplicació està formada per un conjunt de tasques, que seran executades indivisiblement per una única unitat de còmput, és a dir, un worker.

# Capítol 4

# 4 El Problema del Logaritme Discret

En aquest capítol s' exposa el problema que aquest treball estudia: el problema del logaritme discret sobre el grup multiplicatiu $\mathbb{F}_p^*$. A més, s' exposa el funcionament del xifrat elGamal, basat en el problema del logaritme discret, i els possibles atacs que planteja el GDLP.

## 4.1 Logaritme Discret

Sigui $(G, *)$ un grup cíclic finit amb $n$ elements, $*$ l' operació interna de multiplicació, $\alpha$ un generador de $(G, *)$ i un element qualsevol $\beta \in G$, s' anomena *logaritme discret de $\beta$ en base $\alpha$*, i es denota per $\log_\alpha \beta$, a l' únic enter x, $0 \leq x \leq n-1 : \alpha * \overset{x)}{\ldots} * \alpha = \beta$.
Així doncs, el *problema del logaritme discret* generalitzat (*GDLP*) consisteix en:

Donat un grup cíclic finit $(G, *)$ d' ordre $n$, un generador $\alpha \in (G, *)$ i un element $\beta \in G$, trobar l' enter $x$, $0 \leq x \leq n-1$, tal que

$$\alpha * \overset{x)}{\ldots} * \alpha \quad = \quad \beta.$$

Per tal que el GDLP tingui un ús útil en criptografia es requereixen dues condicions: el logaritme discret (característiques dels paràmetres que intervenen en el problema) ha de ser difícil de calcular i l' operació $*$ del grup ha de ser ràpida d' executar. A més, quan l' ordre del grup $G$ té un factor primer molt gran, existeixen grups on el millor algorisme conegut per a la seva ressolució té un cost no polinomial, fet que converteix aquests grups adequats per al seu ús en criptografia.
En [1] s' explica com la dificultat de resoldre el GDLP és independent del generador del grup que es tracta.
Els grups de més interès en criptografia són el grup multiplicatiu $\mathbb{F}_p^*$ d' un cos finit $\mathbb{F}_p$, incloent els casos particulars del grup multiplicatiu $\mathbb{Z}_p^*$ d' enters mòdul un primer $p$, i el grup multiplicatiu $\mathbb{F}_{2^m}^*$ del cos finit $\mathbb{F}_{2^m}$ de característica 2. També d' interès són els grup d' unitats $\mathbb{Z}_n^*$ on $n$ és un enter compost, el grup de punts en una corba el·líptica sobre un cos finit, i el jacobià d'una corba hiperel·líptica definida sobre un cos finit.

## 4.2 Logaritme Discret sobre el grup multiplicatiu $\mathbb{F}_p^*$

El Problema del Logaritme Discret (DLP) es tracta d' una instància del GDLP, en el qual $G = \mathbb{F}_p^* = \{1, 2, \ldots, p-1\}$, on $p$ és un nombre primer i

l' ordre de $\mathbb{F}_p^*$ és $p-1$, de forma que:

> Donat un generador $\alpha \in \mathbb{F}_p^*$ i un element $\beta \in \mathbb{F}_p^*$, el problema és trobar l' enter $x$ , $0 \leq x \leq p-2$, tal que $\alpha^x \equiv \beta \pmod{p}$.

L' operació del grup $\mathbb{F}_p^*$ és la multiplicació mòdul $p$ . Així també, GDLP pot instanciar-se emprant com a grup cossos de la forma $\mathbb{F}_{p^m}$ amb $p$ primer i $m >1$.

Com a exemple d' ús del DLP, en el protocol Diffie-Hellman bàsic, l' emissor $A$ i el receptor $B$ volen compartir un missatge secret. Els passos que es donen en la comunicació són:

1. $A$ genera un primer $p$ , generador $\alpha \in \mathbb{F}_p^*$, $2 \leq \alpha \leq p-2$, un nombre aleatori secret $x$, $1 \leq x \leq p-2$ i calcula $\alpha^x \bmod p$.
2. $A$ envia a $B$ el missatge composat per $p, \alpha$ i $\alpha^x \bmod p$.
3. Un cop $B$ rep el missatge d'$A$, $B$ també tria un nombre aleatori secret $y$, $1 \leq y \leq p-2$, calcula $\alpha^y \bmod p$.
4. $B$ envia a A el missatge $\alpha^y \bmod p$.
5. $B$ calcula el missatge secret elevant $\alpha^x$ al seu nombre secret $y$: $(\alpha^x)^y \bmod p$.
6. $A$ calcula el missatge secret elevant $\alpha^y$ al seu nombre secret $x$: $(\alpha^y)^x \bmod p$.

Tant $A$ com $B$ obtenen el mateix valor, és a dir, el missatge secret compartit.

Malgrat el problema real amb què es trobi un atacant és trobar $\alpha^{xy} \bmod p$, és a dir, trobar els valors aleatoris privats $x$ i $y$, es creu que la ressolució d' aquest problema té la mateixa complexitat que la del DLP.

Altres criptosistemes han estat proposats la seguretat dels quals residia en el DLP, un dels quals és el criptosistema elGamal. En el següent apartat s' exposa el seu mecanisme.

## Criptosistema ElGamal

L' any 1985, T. ElGamal va publicar un criptosistema de clau pública basat en l' exponenciació discreta sobre un grup multiplicatiu $\mathbb{Z}_p^*$ en què $p$ és un primer: el *criptosistema ElGamal*. El seu funcionament es pot dividir en 3 parts:

### Generació de claus

1. Es tria un grup finit $G$ i un element $\alpha \in G$, que no ha de ser necessàriament generador del grup.
2. Cada usuari tria un nombre aleatori $x$, que serà la seva clau privada, i calcula $\alpha^x$ dins de $G$, que serà la seva clau pública.

**Xifratge**

Si l' usuari $A$, amb clau privada $a$, vol enviar un missatge $m \in G$ a un usuari $B$, amb clau privada $b$, llavors realitza el següent:

1. $A$ genera un nombre aleatori $\nu$ i calcula $\alpha^{\nu}$en $G$.
2. $A$ mira la clau pública de $B$, $\alpha^b$, i calcula $\left(\alpha^b\right)^{\nu}$ i $m \cdot \alpha^{b\nu}$ en $G$.
3. $A$ envia a $B$ el parell $\left(\alpha^{\nu}, m \cdot \alpha^{b\nu}\right)$.

**Desxiframent**

Per a recuperar el missatge original, l' usuari $B$ ha de fer el següent:

1. $B$ calcula $\left(\alpha^{\nu}\right)^b$en $G$.
2. $B$ obté el missatge $m$ dividint la segona part del criptograma:

$$m = \frac{m \cdot \alpha^{\nu b}}{\alpha^{\nu b}}$$

Així doncs el problema d' *ElGamal* consisteix en calcular el missatge $m$ coneixent les següents dades: $G$, $\alpha$, $\alpha^a$, $\alpha^b$, $\alpha^v$ i $m \cdot \alpha^{\nu b}$.
La dificultat d' obtenir una clau privada a partir d' una clau pública es basa en la dificultat de resoldre el càlcul del logaritme discret $b = \log_{\alpha} \alpha^b$.

## 4.3 Possibles atacs al GDLP

Aquest apartat es dedica a realitzar una anàlisi breu dels possibles atacs que existeixen al problema del logaritme discret. Per cada algorisme es dóna el temps estimat per resoldre el problema i l' espai que necessita en memòria. Tots ells treballen sobre un grup $G$ d' ordre $n$. Els seus algorismes poden trobar-se en [1].

### 4.3.1 Cerca exhaustiva

Es tracta d' un simple algorisme que opera emprant la força bruta.

**Algorisme 1** Cerca exhaustiva

Entrada: Grup $G$, ordre $n$, generador $\alpha$, element $\beta$
Sortida: Logaritme Discret $t$

1. $t \leftarrow 1$;
2. mentre $t \leq n$ fer
3. $x \leftarrow \alpha^t \bmod n$;
4. si $x = \beta$ llavors
5. retornar *t;*
6. si no llavors
7. $t \leftarrow t + 1$;
8. fi si
9. fi mentre
10. retornar *Error;*

L' objectiu és l' obtenció d' un valor $t$ tal que $\alpha^t$ sigui igual a l' element escollit $\beta$. Una opció és realitzar el càlcul de cada element $\alpha^t$ , des de $t = 1$ fins a l' ordre de $G$. Aquesta opció comporta una pèrdua de temps molt gran. Una opció més adequada seria obtenir el valor $\alpha^t$ a partir de l' anterior:

$$\alpha^t = \alpha^{t-1} \cdot \alpha$$

Malgrat aquesta millora en la rapidesa del càlcul, per calcular nombre grans continua essent un algorisme lent, per tant, és necessari la cerca d' algorismes més efectius.

**Característiques de l' algorisme**

**Temps:** ***O(n)*** **Espai :** ***O(1)***

**Consideracions: Algorisme determinista**

### 4.3.2 Baby-step Giant-Step

Es tracta d' un algorisme que esdevé solució de compromís entre el temps i l' espai d' ús de l' algorisme de l' apartat 4.3.1. Fou creat per Daniel Shanks i es troba englobat en el conjunt de problemes de la classe *arrel quadrada* o *trobada a mig camí.*

**Característiques de l' algorisme**

**Temps:** ***O($\sqrt{n}$)*** **Espai :** ***O($\sqrt{n}$)***

**Consideracions: Algorisme determinista**

### 4.3.3 Pollard Lambda

També anomenat algorisme del *cangur de Pollard*, fou introduit l' any 1978[11] per J. M. Pollard. Es tracta d' un algorisme més ràpid en situacions en les que es coneix d' antuvi que el logaritme cercat es troba en el subinterval $[0, b]$ de $[0, n-1]$, on $b < 0,39n$ [8].

**Característiques de l' algorisme**

$w$ és la longitud de l' interval en què es troba el *logaritme discret.*

**Temps:** $O(\sqrt{w})$ **Espai :** $O(\log w)$

**Consideracions: Algorisme probabilístic.**

**Pot paral·lelitzar-se amb una acceleració lineal.**

### 4.3.4 Pohlig-Hellman

Aquest algorisme, també anomenat *algorisme Silver–Pohlig–Hellman*, treballa bé dins els grups l' ordre del qual no té cap factor primer gran, és a dir, treballa en grups d' ordre *smooth*[12].

**Característiques de l' algorisme**

$n = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$. Cada $p_i$ és un nombre primer diferent.

**Temps:** $O(\sum_{i=1}^{k} e_i \cdot (\log n + \sqrt{p_i}))$ **Espai :** $O(1)$

### 4.3.5 Index Calculus

Es tracta d' un dels algorismes[13] que exploten mètodes per representar els elements del grup com a productes d' elements d' un sistema relativament petit. En aquest mètode, es calcula una base de dades de nombres primers petits i els seus logaritmes corresponents. Això serveix per a poder calcular eficientment logaritmes d' elements arbitraris del grup.
L' algorisme Index Calculus i totes les seves variants, com l' algorisme de Coppersmith i el Number Field Sieve, adaptat per logaritmes, tenen un temps d' execució sub-exponencial.

[11] J.M.Pollard va publicar aquest algorisme aleatori juntament amb el seu més conegut algorisme per resoldre el problema del logaritme discret: la Rho de Pollard.
[12] Un número és $B$-smooth si tots els seus factors primers son $\leq$B.
[13] La idea d' aquest algorisme té lloc el 1922 amb M. Kraitchik, fou recuperada el 1977 per R. Merkle i presentada com la coneixem avui en dia per Leonard Max Adleman.

**Característiques de l' algorisme** [14]

**Temps:** $L_n(\frac{1}{2}, \sqrt[3]{\frac{64}{9}}) = (\sqrt[3]{\frac{64}{9}} + O(1))log^{\frac{1}{3}}n\, loglog^{\frac{2}{3}}n$

**Espai :** $L_n(\frac{1}{2}, c) = (c + O(1))log^{\frac{1}{2}}nlog^{\frac{1}{2}}n$, ***amb una c petita.***

**Consideracions: Algorisme probabilístic.**

**Les seves variants són fàcilments paral·lelitzables.**

[14] L' *L-notació* és una notació asimptòtica, anàloga a la notació de Landau $O$, i denotada com $L_n[\alpha, c]$, per a una variable $n$ que tendeix a l' infinit. $c$ és una constant positiva i $0 \leq \alpha \leq 1$.

# Capítol 5

## 5 Algorisme Rho de Pollard

Entre els atacs més importants al problema del logaritme discret es troba l' algorisme de la Rho de Pollard. És aquest algorisme el que ocupa aquest capítol i representa l' objectiu d' aquest treball.
John M. Pollard va proposar l' any 1978 un interessant algorisme per a resoldre el problema del logaritme discret basat en un mètode de Monte Carlo[15] i el va anomenar *mètode rho*.
El *mètode rho* funciona, en primer terme, definint una seqüència d' elements que serà periòdicament recurrent. Durant la generació d' aquesta seqüència es cerca una coincidència. Aquesta coincidència porta a la solució del problema del logaritme discret amb una alta probabilitat. Les claus d' aquest algorisme són la funció iterativa per a generar la seqüència i l' algorisme per a la cerca de cicles per a detectar la coincidència.
L' algorisme Rho de Pollard està basat en la paradoxa de l' aniversari: si escollim a l' atzar elements d' un sistema d' $n$ elements, es necessita prop de $\sqrt{n}$ elements fins aconseguir un element dos cops (coincidència o col·lisió).

**¿ Per què s' anomena Rho l' algorisme ?**

El conjunt de valors que s' obtenen es troben situats dins d' un conjunt que es defineix com a finit. Veure la definició 3.1.2.2.
És així com en la generació de valors es produeix, en un temps qualsevol, una repetició numèrica, una col·lisió. En representar els valors per punts la gràfica resultant presenta una forma similar al que té la lletra grega $\rho$, tal com es pot veure en la Figura 1. L' origen del nom rho de Pollard prové d' aquesta idea.

### 5.1 Rho de Pollard

L' algorisme Rho de Pollard genera una successió d' elements amb la finalitat de trobar una igualtat numèrica que s' utilitza no només per a resoldre el problema del logaritme discret sinó també per a altres problemes matemàtics.
La complexitat d' aquest algorisme és la mateixa que la del Baby-step Giant-step però té l' avantatge que no necessita reservar espai en memòria, és a dir, té un cost espaial *O(1)*.
A continuació, s' estudia el mètode de rho sobre el grup multiplicatiu $\mathbb{F}_p$.
El procediment del *mètode de rho*:

[15] Els mètodes de Monte Carlo abarquen una col·lecció de tècniques que permeten obtenir solucions de problemes matemàtics o físics per mitjà de proves aleatòries repetides.

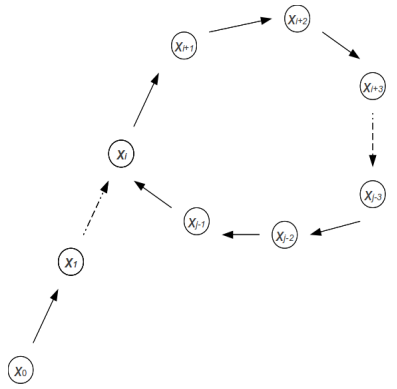


Figura 1: Cicle de la $\rho$ de Pollard

Suposem un grup $(G, *)$ cíclic d' ordre $n$, un generador $\alpha$ i un element $\beta$.
L' objectiu és calcular $\log_\alpha \beta$.
L' algorisme empra la següent funció o aplicació:

$$\begin{array}{rccc} f: & G & \longrightarrow & G \\ & (x_i, a_i, b_i) & \longmapsto & (x_j, a_j, b_j), \end{array}$$

tal que $x_i = \alpha^{a_i} \cdot \beta^{b_i} \in G$, $(x_j, a_j, b_j) = f(x_i, a_i, b_i)$ i $x_j = \alpha^{a_j} \cdot \beta^{b_j} \in G$. Ja que la funció $f$ ha de tenir un comportament pseudoaleatori, Pollard suggereix dividir els elements de $G$ en tres conjunts diferents ($C_1$, $C_2$ i $C_3$), per exemple, emprant la següent funció:

$$d(x) = \begin{cases} x \in C_1 & \text{si } f(x) \equiv 1 \pmod 3, \\ x \in C_2 & \text{si } f(x) \equiv 0 \pmod 3, \\ x \in C_3 & \text{si } f(x) \equiv 2 \pmod 3. \end{cases}$$

A continuació, es defineix la funció $h$ següent, la qual té un comportament molt semblant al que tindria una funció aleatòria.

$$x_{i+1} = h(x_i) = \begin{cases} \beta * x_i & \text{si } x_i \in C_1 \quad , \\ x_i * x_i & \text{si } x_i \in C_2 \quad , \\ \alpha * x_i & \text{si } x_i \in C_3 \quad . \end{cases}$$

Aquesta successió d' elements $x_i$ defineix alhora un parell de successions més, les quals seran d' enters mòdul $n$: les successions $a_i$ i $b_i$. Amb aquests elements, es pot construir una funció $F$ que retorni una tripleta a partir de l' anterior de la següent forma,

$$F(x_i, a_i, b_i) = \begin{cases} (\beta * x_i, a_i, b_i + 1 \pmod{n}) & \text{si } x_i \in C_1 \quad , \\ (x_i * x_i, 2a_i \pmod{n}, 2b_i \pmod{n}) & \text{si } x_i \in C_2 \quad , \\ (\alpha * x_i, a_i + 1 \pmod{n}, b_i) & \text{si } x_i \in C_3 \quad . \end{cases}$$

Els valors $a_0$ i $b_0$ de la tripleta inicial no és necessari inicialitzar-los a cap valor inicial, l' algorisme implementat s' adapta a aquest fet i inicialitza els paràmetres $a_0$ i $b_0$ a valors aleatoris entre 1 i $p-1$, tot respectant la igualtat

$x_i = \alpha^{a_i} * \beta^{b_i}$ . En cas de la primer tripleta, $x_0 = \alpha^{a_0} * \beta^{b_0}$.

**¿ Com obtenim el logaritme discret ?**

L' objectiu és trobar uns índexs $i$ i $j$, de manera que $x_i = x_j$ i $i \neq j$. En cas de trobar-los es té

$$x_i = \alpha^{a_i} * \beta^{b_i} \ , \ x_j = \alpha^{a_j} * \beta^{j}.$$

Apliquem el logaritme en base $\alpha$ a les dues expressions:

$$\log_\alpha(x_i) \equiv \log_\alpha(\alpha^{a_i} * \beta^{b_i}) \pmod{n},$$

$$\log_\alpha(x_j) \equiv \log_\alpha(\alpha^{a_j} * \beta^{b_j}) \pmod{n}.$$

Operem el logaritme d' una multiplicació en ambdues expressions:

$$\log_\alpha(x_i) \equiv a_i + b_i \log_\alpha(\beta) \pmod{n},$$

$$\log_\alpha(x_j) \equiv a_j + b_j \log_\alpha(\beta) \pmod{n}.$$

Restem ambdues expressions:

$$(b_i - b_j) \ \log_\alpha(\beta) \equiv (a_j - a_i) \pmod{n}.$$

Per últim, passem dividint el factor $(b_i - b_j)$. Obtenim:

$$\log_\alpha(\beta) \equiv (a_j - a_i)\,(b_i - b_j)^{-1} \pmod{n},$$

Aquest darrer pas resulta clau per a l' obtenció, o no, del logaritme discret. Si $b_i \equiv b_j$, el terme $(b_i - b_j)$ no té invers a $\mathbb{Z}_p$ i l' equació no pot ser resolta. Altrament, s' obté el logaritme discret $\log_\alpha(\beta)$.

Tal com s' ha explicat en aquest apartat, la ressolució del problema mitjançant l' algorisme Rho de Pollard implica la detecció d' un cicle en una seqüència trobant una col·lisió de valors numèrics. Per a trobar un cicle dins una seqüència donada existeixen bàsicament dos algorismes, que s' exposen a continuació.

## 5.2 Algorisme de Floyd

Un dels algorismes per a la cerca de cicles és l' algorisme de Floyd (Robert W. Floyd, 1976), també conegut com *algorisme de la "tortuga i la llebre"*. L' algorisme empra dues tripletes que recorren el grup sobre el que treballen. Inicialment l' algorisme contempla una tripleta $(x_0, a_0, b_0)$. La tripleta que representa la tortuga aplica la funció $F(x_i, a_i, b_i)$ un cop per iteració (avança una passa), la tripleta que representa la llebre aplica la funció $F(x_i, a_i, b_i)$ dos cops per iteració (avança dues passes). Aquest algorisme presenta el següent procediment:

**Algorisme 2** Algorisme de Floyd

Entrada: Tripleta inicial$(x_0, a_0, b_0)$
Sortida: Dues tripletes$(x_t, a_t, b_t)$, $(x_l, a_l, b_l)$, on $x_t \neq x_l$

1. $(x_t, a_t, b_t) \leftarrow (x_0, a_0, b_0)$;
2. $(x_l, a_l, b_l) \leftarrow F(x_t, a_t, b_t)$;
3. mentre $x_t \neq x_l$ fer
4. $\quad (x_t, a_t, b_t) \leftarrow F(x_t, a_t, b_t)$;
5. $\quad (x_l, a_l, b_l) \leftarrow F(F(x_l, a_l, b_l))$;
6. fi mentre
7. retornar $(x_t, a_t, b_t)$, $(x_l, a_l, b_l)$;

Considerant $\lambda$ la longitud de la cua de la successió, $\mu$ la longitud del cicle i $(x_i = x_{2i})$ la col·lisió, podem dir que l' índex de col·lisió $i$ és:

$$i = \mu \left(1 + \left\lfloor \frac{\lambda}{\mu} \right\rfloor\right)$$

**Característiques de l' algorisme**

**Nombre d' operacions:** $3\sqrt{p}$ **Espai :** ***O(1)***

A continuació, s' exposa un exemple de problema de logaritme discret sobre un grup multiplicatiu $\mathbb{F}_p^*$.

Els valors inicials del problema són:

$$\mathrm{G} = \mathbb{F}^*_{2549}, \qquad \alpha = 2 \quad \text{i} \quad \beta = 1234$$

Tal com s' observa en la taula 1, l' algorisme ha trobat una col·lisió: $x_{77} = x_{154} = 581$. En primer terme, es calcula

$$b = b_{77} - b_{154} = 339.$$

En segon terme, s' inverteix $b$ en el grup $G$:

$$b^{-1} = 391 \pmod{2548}$$

L' operació de trobar l' invers de $b$ pot no ser exitosa, tal com s' ha exposat en l' apartat anterior. Si $b$ no té invers (consultar proposició 3.1.3.3), l' algorisme retorna *error* i torna a executar-se amb variables inicials aleatòries. En cas que $b$ sigui invertible en $G$, es calcula el logaritme discret:

$$\log_\alpha \beta = b^{-1}(a_{154} - a_{77}) = 2382.$$

| $i$ | $x_i$ | $a_i$ | $b_i$ | $x_{2i}$ | $a_{2i}$ | $b_{2i}$ |
|---|---|---|---|---|---|---|
| 1 | 1584 | 585 | 1645 | 840 | 1170 | 742 |
| 2 | 840 | 1170 | 742 | 1966 | 2132 | 420 |
| 3 | 2076 | 2340 | 1484 | 1764 | 2132 | 422 |
| 4 | 1966 | 2132 | 420 | 1283 | 1717 | 844 |
| 5 | 2023 | 2132 | 421 | 34 | 1719 | 844 |
| 6 | 1764 | 2132 | 422 | 809 | 1720 | 845 |
| 7 | 1916 | 1716 | 844 | 2531 | 1721 | 846 |
| 8 | 1283 | 1717 | 844 | 2477 | 1723 | 846 |
| 9 | 17 | 1718 | 844 | 2261 | 1725 | 846 |
| 10 | 34 | 1719 | 844 | 1397 | 1727 | 846 |
| ... | ... | ... | ... | ... | ... | ... |
| 67 | 411 | 1968 | 1082 | 2026 | 2083 | 353 |
| 68 | 687 | 1388 | 2164 | 1906 | 2083 | 355 |
| 69 | 404 | 228 | 1780 | 1622 | 2083 | 357 |
| 70 | 808 | 229 | 1780 | 1390 | 2085 | 357 |
| 71 | 1516 | 229 | 1781 | 2440 | 2085 | 359 |
| 72 | 43 | 229 | 1782 | 1014 | 1622 | 720 |
| 73 | 699 | 229 | 1783 | 607 | 696 | 1441 |
| 74 | 1742 | 458 | 1018 | 332 | 696 | 1443 |
| 75 | 935 | 459 | 1018 | 1902 | 697 | 1444 |
| 76 | 1870 | 460 | 1018 | 2057 | 240 | 680 |
| 77 | **581** | 460 | 1019 | **581** | 242 | 680 |

Taula 1: Rho de Pollard sobre el grup multiplicatiu $\mathbb{F}^*_{2549}$

## 5.3 Algorisme de Brent

L' algorisme de Brent (Richard P. Brent) per a la cerca de cicles esdevé una versió de l' algorisme de Floyd més ràpida. L' algorisme de Brent també empra els noms de *llebre* i *tortuga* per a referir-se a les dues tripletes que avancen en la seqüència de valors dins el grup corresponent. En aquest algorisme, la llebre executa la funció $F(x_i, a_i, b_i)$ un cop (avança una passa) per iteració mentre la tortuga roman immòbil. La tortuga pren el valor de la llebre un cop aquesta ha efectuat $2^i$ passes, és a dir, quan la tripleta que representa la llebre ha aplicat la funció $F(x_i, a_i, b_i)$ $2^i$ cops.

**Algorisme 3** Algorisme de Brent

Entrada: Tripleta inicial$(x_0, a_0, b_0)$
Sortida: Dues tripletes$(x_t, a_t, b_t)$, $(x_l, a_l, b_l)$, on $x_t = x_l$

1. $(x_t, a_t, b_t) \leftarrow (x_0, a_0, b_0)$;
2. $(x_l, a_l, b_l) \leftarrow F(x_t, a_t, b_t)$;
3. $aturada \leftarrow 1$;
4. $passes \leftarrow 1$;
5. mentre $x_t \neq x_l$ fer
6. si $passes = aturada$ llavors
7. $(x_t, a_t, b_t) \leftarrow (x_l, a_l, b_l)$;
8. $aturada \leftarrow 2 \cdot aturada$;
9. $passes \leftarrow 0$;
10. fi si
11. $(x_l, a_l, b_l) \leftarrow F(x_l, a_l, b_l)$;
12. $passes \leftarrow passes + 1$;
13. fi mentre
14. retornar $(x_t, a_t, b_t)$, $(x_l, a_l, b_l)$;

En [9] Brent defensa que l' ús del seu algorisme de cerca de cicles és un 36 % més ràpid que l' algorisme de Floyd i que augmenta la velocitat de l' algorisme Rho de Pollard al voltant d' un 24 %.

**Característiques de l' algorisme**

**Nombre d' operacions:** $3\sqrt{p}$ **Espai : O(1)**

## 5.4 Paral·lelització de l' algorisme original

En aquest apartat, s' expliquen els diferents mètodes que s' han dut a terme per a la paral·lelització de l' algorisme de la Rho de Pollard i que han portat a disminuir el temps de càlcul, i per tant d' execució, del mateix algorisme per a diferents problemes del logaritme discret.
És de ressaltar el caràcter seqüencial de l' algorisme original, és a dir, pot ser executat de forma lineal. Les propostes que a continuació s' exposen permeten millorar l' algorisme Rho de Pollard clàssic.

### 5.4.1 Paral·lelització trivial

Aquesta versió de paral·lelització de l' algorisme esdevé la més senzilla de totes. Aquest mètode consisteix en el llançament de l' algorisme, amb una tripleta inicial diferent en cada cas, en cadascun dels processadors (o workers) de forma independent. Un cop un dels processadors ha trobat el logaritme discret sobre el grup multiplicatiu indicat, retorna el resultat i els demés processadors aturen el seu càlcul.

**Característiques de la paral·lelització tenint P processadors**

**Nombre d' operacions:** $3\sqrt{\frac{p}{P}}$ **Espai :** ***negligible***

El problema de paral·lelitzar l' algorisme amb aquest mètode és que l' *speedup*[16] que s' obté presenta només un factor de $\sqrt{P}$. Aquest factor quadràtic és força ineficient en tant que es necessiten $P$ processadors per a reduir el temps de càlcul una $\sqrt{P}$ part.

### 5.4.2 Paral·lelització basada en la cerca de cicles per mitjà de punts distingits

En [8], M.J. Wiener i P.C. Van Oorschot presenten una versió de l' algorisme original que proporciona un *speedup* de $P$ si es disposen de $P$ processadors. Aquest algorisme empra una zona de memòria on s' emmagatzemen les tripletes distingides.
El procediment que es duu a terme en aquesta proposta es basa fonamentalment en la paral·lelització de l' apartat anterior. El processador central llença l' algorisme o procediment a cadascun dels processadors *workers*. Així doncs, cada processador construeix la seva seqüència de tripletes fins trobar-ne una, tripleta o punt distingit, que compleix una certa condició de fàcil comprovació[17]. El processador que ha trobat una tripleta distingida $(x_d, a_d, b_d)$ l' envia al servidor o processador central, i aquest l' emmagatzema, i comença l' algorisme de nou amb variables inicials diferents. D' aquesta manera, el processador central va rebent tripletes distinguides i les va emmagatzemant en el seu espai de memòria. L' algorisme finalitza un cop aquest processador rep una nova tripleta distingida $(x_{d'}, a_{d'}, b_{d'})$ tal que $x_{d'}$ coincideix amb el valor $x_d$ d'una tripleta emmagatzemada. Així doncs, es produeix una col·lisió, en aquest cas, no en els *workers* sinó en el processador responsable de rebre les tripletes distingides.

En [8] s'exposa que la proporció d' elements del grup $G$, en què es treballa, que compleixen aquesta condició és $\theta$. El nombre d' operacions, doncs, que realitza cada processador fins trobar una col·lisió és:

$$\frac{1}{P}\sqrt{\frac{\pi p}{2}} + \frac{1}{\theta}$$

[16] L' *speedup* es refereix a quant de ràpid és un algorisme en paral·lel respecte la seva execució seqüencialment. El seu càlcul és el quocient entre els temps d' execució en paral·lel i en sèrie.
[17] En [8], es proposa escollir aquells valors $x_i$ la representació binària del qual tingui un nombre concret de bits a zero en la part baixa.

**¿ Sempre es troben punts distingits ?**

La seqüència de tripletes generades per cadascun dels processadors pot no contenir tripletes distingides. Wiener i van Oorschot proposen el valor de $\frac{20}{\theta}$ com a límit de generació de tripletes. Si aquest nombre de generacions de valors és excedit i, per tant, no s' ha trobat cap punt distingit es provoca l' execució de nou de l' algorisme emprant una nova tripleta inicial.

**¿ S' estableix un límit de mèmoria per a la taula de tripletes ?**

Si es disposen de $P$ processadors i aquests envien tripletes distingides amb una proporció de $\theta$, el nombre de tripletes a emmagatzemar és:

$$\theta\sqrt{\frac{\pi p}{2}} + P$$

La resposta a la pregunta que es planteja és incerta ja que el valor de $\theta$ ens determina el tamany de memòria a més del cost temporal que es veu disminuit o incrementat segons el nombre de tripletes enviades. Així també, en [9] es contempla la situació en què es disposi d' una memòria limitada per emmagatzemar tripletes. Les opcions que es proposen son:

1. Prendre un valor elevat de $\theta$ fins ajustar la fòrmula anterior a les necessitats de memòria.
2. Prendre un valor gran de $\theta$, ajustar el tamany de la taula o *hash* i inserir tripletes fins que la taula estigui plena. En aquest cas, s' afegirien les noves i s' esborrarien les antigues.

Aquesta versió de paral·lelització proporciona un *speedup* lineal que resulta eficient. A més, aquest procediment necessita d' una zona de memòria prou gran per tal de no penalitzar el cost temportal de l' algorisme.

# Capítol 6

# 6 Disseny i implementació

Aquest capítol conté la informació de quins han estat els procediments emprats per arribar a l' objectiu d' aquest treball, quines han estat les eines de software emprades i sobre quin hardware s' ha treballat. A més, s' exposaran conceptes fonamentals, que no s' han explicat en aquest document encara, de la plataforma sobre la que s' ha dut a terme aquest projecte: CoDiP2P. En darrer terme es detalla el programari emprat per a la confecció d' aquesta documentació.

## 6.1 El llenguatge de programació

El codi de la implementació de la Rho de Pollard s' ha escrit amb el llenguatge interpretat d' alt nivell Java, sobre l' entorn de programació (IDE) Netbeans 7.0.
Aquesta decisió es va prendre arran de saber que la plataforma CoDiP2P està implementada en llenguatge Java. Si el codi hagués estat escrit en un altre llenguatge s' hauria d' haver convertit al llenguatge sobre el que treballa CoDiP2P: Java. Precisament, un dels punts forts del sistema peer-to-peer CoDiP2P és el seu ús en un màxim nombre d' usuaris. Quant més usuaris hi ha, el sistema ofereix una capacitat de còmput major. Aquest fet és possible si es duu a terme una aplicació que es pugui executar en arquitectures o entorns de treballs diferents. Per aquest motiu, es va optar per la implementació de la plataforma amb llenguatge Java.

### 6.1.1 Llibreria BigInteger

La implementació de la Rho de Pollard implica l' ús de nombres grans. Java ens proporciona l' ús de tipus *int* i *long* que emmagatzemen nombre de 32 i 64 bits cadascun. Les limitacions d' aquests tipus de dades per a problemes d' entitat han plantejat l' ús d'alguna llibreria que tractés nombres més grans i mètodes associats a ells. Java presenta la llibreria BigInteger (*java.math.BigInteger*) que tracta nombres de major longitud. A més, presenta analogies a tots els operadors Java sobre els enters, hereda tots els mètodes importants de la llibreria fonamental d' operacions matemàtiques *java.lang.Math* i, a més, proveeix operacions per l' aritmètica modular, el càlcul del màxim comú divisor, test de primalitat, generació de nombres primers, manipulació de bits, etcètera.

## 6.2 El maquinari

Els computadors emprats en l' execució del programa de què tracta aquest treball presenten les següents característiques:

**Pentium 4 biprocessador a 2.66GHz, 2048 MB de memòria RAM i una cache de 128 kB.**

## 6.3 Plataforma CoDiP2P

Per tal d' emprar la plataforma CoDiP2P, s' ha utilitzat la partició *Fedora Compdist* de tots els nodes de la xarxa. Un cop coneguts els noms d' usuari i contrassenyes de tots els computadors, podem efectuar cadascuna de les operacions que es detallen a continuació.
Els passos que s' han dut a terme per preparar la plataforma CoDiP2P a l' execució de l' algorisme són els següents:

1. La comunicació entre els nodes workers i el node NMajor es produeix copiant la clau pública del node NMajor als demés nodes workers, i viceversa. La generació de les claus es duu a terme amb la comanda *ssh-keygen* amb l' opció de tipus de clau que es genera: *-t rsa*.

2. Es realitza una còpia de la carpeta principal, que conté el codi de la plataforma CoDiP2P, un intèrpret de la plataforma des de línia de comandes i la carpeta que correspon al projecte de què tracta aquest treball: PollardRhoP2P, a tots els nodes que intervenen en l' execució del programa.

3. Es crea la jerarquia de nodes. En aquest tercer pas s' executa el codi corresponent a la plataforma CoDiP2P en cadascun dels peers de la xarxa. El següent pas és definir qui és el node NMajor de la jerarquia

```
[codip2p@eps34-18 CoDiP2P]$ java -classpath ./lib -jar ./CoDiP2P.jar  --nmajor
```

Figura 2: Declaració del node NMajor

   i notificar-ho a tots els demés nodes de la xarxa, indicant l' adreça IP del node NMajor.

```
[codip2p@eps34-19 CoDiP2P]$ java -classpath ./lib -jar ./CoDiP2P.jar
 --seed tcp://172.16.2.147:9700
```

Figura 3: Notificació de NMajor a un node de la xarxa

El node NMajor permet l' accés al sistema dels diferents peers de la xarxa. Aquest node, que es troba en execució, mostra per pantalla el conjunt de peers workers preparats a executar les tasques.

```
08/09/2011 21:06:31 codip2p.platform.services.maintenance.AreaMaintenanceThread printWorkersTable
INFO: WORKERS TABLE:
urn:jxta:uuid-59616261646162614E50472050325033EEAE855A661340E4AF8583A11C476EDD03
urn:jxta:uuid-59616261646162614E504720503250336FF9D656B02947088513DFE9746D104803
urn:jxta:uuid-59616261646162614E50472050325033D11F2399D5104FB3A5C5AC0FEBED595B03
urn:jxta:uuid-59616261646162614E504720503250339 7A08FCC5B5E405DBB50EA6153C240B503
urn:jxta:uuid-59616261646162614E50472050325033DA8B1542B81D461E9FED8858DB2AB1B403
urn:jxta:uuid-59616261646162614E50472050325033FFF89410D2EE43098F10EFB25201EB7103
urn:jxta:uuid-59616261646162614E50472050325033A3C211E4E74D4D3B98E6B2E8E53F83E503
urn:jxta:uuid-59616261646162614E504720503250334 6BEF6E5AD204B13A4C3F30DC261408C03
urn:jxta:uuid-59616261646162614E50472050325033CEB4F1AA0C2A48FA8A59DE089F1066BB03
urn:jxta:uuid-59616261646162614E504720503250338C21BB083416438BAA749F77AB94E4C303
urn:jxta:uuid-59616261646162614E504720503250338 7B0AA614E8849588FD9CB3EA7998B8503
urn:jxta:uuid-59616261646162614E50472050325033A1E64C2D998148358756AADD6393D25103
```

Figura 4: Taula de peers workers

Un cop s' han afegit tots els peers workers al sistema el missatge que es mostra per pantalla és el que es mostra en la Figura 5.

```
08/09/2011 21:06:34 codip2p.network.jxta.JXTAMessagesModule pipeMsgEvent
INFO: All listenners were succesfully notified
```

Figura 5: Missatge d' avís de peers preparats

4. S' executa l' aplicació Rho de Pollard en cadascun dels nodes amb els paràmetres corresponents del problema: grup multiplicatiu $\mathbb{F}_p^*$, generador del grup $\alpha$ i el nombre de tasques que s' executen.

Cadascun d' aquests passos s' ha dut a terme gràcies a fitxers scripts[18] que de manera reiterada han executat aquests processos a cadascuna de les màquines o nodes.

### ¿ Com s' executa l' aplicació ?

La distribució de les tasques en els nodes de la xarxa s' efectua de la següent forma:

[18] Un *script* és un programa que s' emmagatzema en un fitxer de text pla i l' ús habitual de les ordres que conté és realitzar diverses tasques com, per exemple, interactuar amb el sistema operatiu o amb l' usuari.

1. L' execució del programa s' efectua en un dels peers workers. La Figura 6 mostra el nombre de tasques que s' envien: 19. La idea consisteix en què el sistema distribueix les 19 tasques en els 19 peers workers que intervenen en la cerca del logaritme discret.

```
[codip2p@eps34-19 CoDiP2P]$ java -classpath ./lib -jar
./PollardRhoP2P/dist/PollardRhoP2P.jar 492930551 7 19
--jarPath ./PollardRhoP2P/dist/PollardRhoP2P.jar
```

Figura 6: Execució del programa

2. Si la comunicació a un peer worker en l' assignació d' una tasca es perd, el sistema s' adapta a aquest fet i reenvia la tasca.
3. Si un peer worker retorna un resultat sense èxit, l' aplicació genera una nova tasca i li és assignada.
4. Si un peer worker retorna un resultat amb èxit, s' atura l' execució del programa i s' espera la finalització de les tasques dels demés peers.

## 6.4 LaTeX

TeX és un sistema de tipografia desenvolupat per Donald Ervin Knuth, el qual és molt popular en l' ambient acadèmic, especialment entre les comunitats de matemàtics, físics i informàtics.

LaTeX és un conjunt de macros de TeX que facilita l' ús d' aquest. La idea principal de LaTeX és ajudar l' usuari que escriu un document a centrar-se en el contingut més que en la forma. LaTeX és programari lliure sota llicència *LPPL*.

LyX és un programa gràfic creat per Matthias Ettrich que permet editar text emprant LaTeX. Es tracta d' un processador de textos que ajuda a no pensar en el format final del treball i a centrar-se en el contingut i la seva estructura. LyX és programari lliure.

## 6.5 Implementació de la Rho de Pollard sobre el grup $\mathbb{F}_p^*$

La implementació de l' algorisme Rho de Pollard s' ha realitzat sobre el grup multiplicatiu $\mathbb{F}_p^*$ . Aquesta secció presenta el pseudocodi dels algorismes relacionats amb aquest treball i les funcions més importants que s' han emprat en la seva implementació.

### 6.5.1 Procediments o funcions més importants

Les funcions més importants que intervenen en l' algorisme implementat són les següents:

- **BigInteger random(BigInteger i)**

  Funció que retorna un element aleatori entre 1 i $i - 1$. Permet escollir el valor de l' element $\beta$ d' un grup qualsevol, que es necessita com a valor d' inici de l' algorisme així com les variables inicials $a$ i $b$.

- **BigInteger solution(BigInteger[ ] x1, BigInteger[ ] x2)**

  Funció que, a partir de les dues tripletes involucrades en la col·lisió x1 i x2, retorna *error* en cas que no existeixi logaritme discret o el *logaritme discret*, en cas que el pugui calcular.

- **BigInteger[] next(BigInteger[ ] i)**

  Funció que retorna la tripleta següent en la seqüència a partir de la tripleta actual $i$. Així doncs, la funció *next* calcula el parell $(x_i, x_{2i})$a partir del parell anterior $(x_{i-1}, x_{2i-2})$.

### 6.5.2 Pseudocodi de la Rho de Pollard

L' algorisme original de la Rho de Pollard té el següent pseudocodi:

**Algorisme 4** Algorisme Rho de Pollard

Entrada: Grup $G$, ordre $n$, generador$\alpha$, element$\beta$
Sortida: Logaritme discret $t$

1. funcio next(x,a,b)
2. $m \leftarrow$ x mod 3;
3. si $m = 0$ llavors
4. $tr \leftarrow (x^2 \pmod{n+1}, 2a \pmod{n}, 2b \pmod{n})$;
5. si no si $m = 1$ llavors
6. $tr \leftarrow (x \cdot \beta \pmod{n+1}, a, b+1 \pmod{n})$;
7. si no llavors
8. $tr \leftarrow (x \cdot \alpha \pmod{n+1}, a+1 \pmod{n}, b)$;
9. retornar *tr;*
10. funcio random(i)
11. retornar $j \in [1, i-1]$;

*programa principal*

12. $a \leftarrow random(n+1)$;
13. $b \leftarrow random(n+1)$;
14. $(x_i, a_i, b_i) \leftarrow (\alpha^a \cdot \beta^b, a, b)$;
15. $(x_j, a_j, b_j) \leftarrow next\,(x_i, a_i, b_i)$;
16. mentre $x_i \neq x_j$ fer
17. $(x_i, a_i, b_i) \leftarrow next\,(x_i, a_i, b_i)$;
18. $(x_j, a_j, b_j) \leftarrow next(next(x_j, a_j, b_j))$;
19. fi mentre
20. si mcd$(b_i - b_j, n) \neq 1$ llavors
21. retorna *"error"*;
22. si no llavors
23. $t \leftarrow (a_j - a_i)\,(b_i - b_j)^{-1} \pmod{n}$;
24. retorna $t$;

### 6.5.3 Pseudocodi de la Rho de Pollard paral·lelitzada

Els algoritmes emprats en l' execució de la rho de Pollard paral·lelitzada han estat dos: l' algorisme 4 i una proposta suportada en el treball de Paul C. van Oorschot and Michael J. Wiener en [8] i exposada en l' apartat 5.4.2. El seu pseudocodi és el següent:

**Algorisme 5** Algorisme Rho de Pollard emprant la cerca de punts distingits per a la cerca de cicles

Entrada: Grup $G$, ordre $n$, generador $\alpha$, element $\beta$
Sortida: Logaritme Discret $t$

1. Inicialitzar taula hash
2. Cada node realitza:
3. comptador $\leftarrow$1;
4. $(x_t, a_t, b_t) \leftarrow (x_0, a_0, b_0)$;
5. $(x_l, a_l, b_l) \leftarrow F(x_t, a_t, b_t)$;
6. mentre contador $\leq$ llargada_maxima_sequencia fer
7. si $(x_t, a_t, b_t)$ o $(x_l, a_l, b_l)$ és un punt distingit llavors
8. retornar punt distingit $(x_t, a_t, b_t)$ o $(x_l, a_l, b_l)$;
9. fi si
10. $(x_t, a_t, b_t) \leftarrow F(x_t, a_t, b_t)$;
11. $(x_l, a_l, b_l) \leftarrow F(F(x_l, a_l, b_l))$;
12. comptador $\leftarrow$comptador +1;
13. fi mentre

El pseudocodi de l' algorisme 5 té com a fonaments els de l' algorisme 4.
L' estructura de memòria que s' ha emprat ha estat una taula *hash*. El motiu principal d' aquesta elecció és l' accés ràpid i directe a aquesta estructura. El tamany del hash s' ha fixat en un nombre primer. Aquesta elecció s' ha produit pel fet que si un hash presenta un tamany d' un valor que té divisors comuns amb el tamany del hash es provoquen col·lisions en la inserció de valors en realitzar la funció mòdul. D' aquesta forma, es contempla que el tamany del hash ha de ser superior al nombre d' elements que insereix. D' aquesta manera, s' eviten acumulacions de valors seguits a la taula. Tal i com s' exposa en [15], el tamany escollit pel hash és el nombre primer següent al producte del nombre de peers workers i 16.
El comptador serveix per anotar el nombre d' iteracions en les que es generen el conjunt de tripletes.
La llargada màxima de la seqüència es refereix al nombre màxim d' iteracions que executa l' algorisme en cas que aquest no trobi cap punt distingit. En [8] s' aconsella el valor $\frac{20}{\theta}$. $\theta$ representa la proporció d' elements que compleixen la propietat de punt distingit. El valor que s' ha pres de $\theta$ ha estat $\frac{1}{32}$. En l' algorisme implementat, una tripleta representa un punt distingit si el seu valor és divisible per 32 (els cinc darrers bits es troben a 0). Per tant, la proporció de punts distingits en qualsevol grup multiplatiu serà $\frac{1}{32}$.

El programa principal va emmagatzemant totes les tripletes que representen punts distingits i efectua contínuament la comprovació $(x_l, a_l, b_l) \neq (x_j, a_j, b_j)$ amb $(x_l = x_j)$ on $(x_l, a_l, b_l)$ és el darrer punt distingit rebut i $(x_j, a_j, b_j)$ cadascuna de les tripletes que es troben en el hash. Si aquesta condició es compleix, intenta trobar el logaritme discret segons s' explica en l' apartat 5.1. Altrament, continua esperant nous punts distingits.

# Capítol 7

# 7 Avaluació experimental i conclusions

El conjunt de proves que s' ha dut a terme s' ha efectuat sobre la plataforma de computació distribuida peer-to-peer CoDiP2P en la que han intervingut 20 computadors biprocessador, dels quals 19 han pres el rol de workers i 1 ha representat el node NMajor, el qual és l' únic node que no ha executat l' aplicació de la Rho de Pollard.

A continuació es mostren els resultats obtinguts per cadascun dels exemples tractats. Cadascun dels problemes s'han identificat pel nombre de dígits que presenta el tamany del grup multiplicatiu sobre el que es realitza l' atac.

## 7.1 Resultats Rho de Pollard sobre la plataforma CoDiP2P

El resultat de cadascuna de les proves es tracta de la mitjana aritmètica de les 5 proves efectuades sobre el mateix grup multiplicatiu $\mathbb{F}_p^*$. A la Taula 2 es mostren els resultats temporals de l' execució de la Rho de Pollard sobre un computador Pentium IV biprocessador a 2.66 GHz i 2048 MB de memòria RAM localment (temps I) i sobre la plataforma CoDiP2P amb 19 processadors (temps II).

| dígits $p$ | temps I (s) | temps II (s) |
|---|---|---|
| 4 | 0.13 | 1.10 |
| 5 | 0.49 | 3.89 |
| 6 | 1.6 | 4.27 |
| 7 | 4.56 | 15.23 |
| 8 | 8.29 | 22.64 |
| 9 | 42.27 | 43.15 |
| 10 | 165.08 | 79.10 |
| 11 | 210.65 | 96.65 |
| ... | ... | ... |

Taula 2: Temps mitjans de la Rho de Pollard sobre CoDiP2P amb l'algorisme implementat

En la Figura 7 podem observar l' SpeedUp corresponent als temps de la Taula 2.

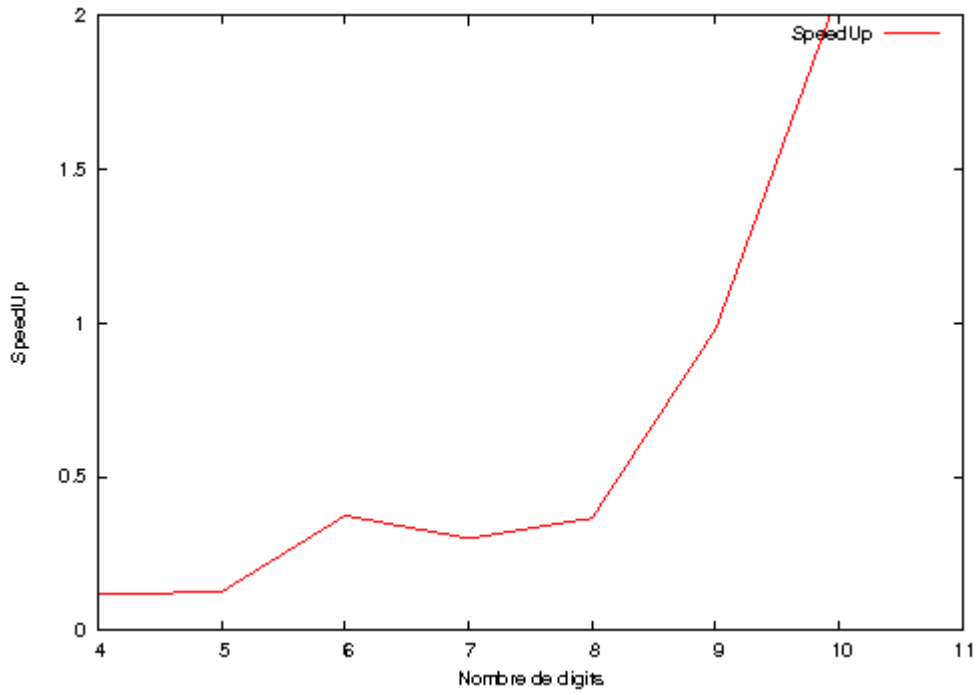


Figura 7: SpeedUp de la Rho de Pollard sobre el grup multiplicatiu $\mathbb{F}_p^*$ en CoDiP2P

L' algorisme emprat ha estat l' algorisme Rho de Pollard basat en la cerca de col·lisions mitjançant punts distingits.
Els resultats obtinguts presenten la següent anàlisi:

L' Speed Up que s' espera de l' algorisme Rho de Pollard respecte la seva execució en local no s' aproxima a l' obtingut en les proves. Els principals motius que s' han estudiat i que poden ser la causa d' aquests resultats tant inesperats són els següents:

1. La plataforma CoDiP2P presenta força inconvenients per la quantitat de comunicacions que es duen a terme. Durant l' execució del programa, els nodes de la xarxa envien constantment resultats i reben noves tasques a executar. En els casos en què el problema és petit, és a dir, es tracten grups el tamany dels quals és petit, és ineficient la seva execució ja que les dades pertanyents a les comunicacions de la plataforma superen amb escreix les dades de la pròpia aplicació. Així també, s' observa com la plataforma CoDiP2P funciona quelcom millor quan es compensa el nombre de dades enviades per la pròpia plataforma, per a la correcta conjunció de tots els elements del sistema, amb el nombre de dades de l' aplicació. No obstant, per a problemes amb nombres primers majors de 30 bits el temps

d' execució és molt gran i la seva mesura, inabastable.

2. La plataforma és força lenta, i en alguns casos ineficient, en l' agregació de nous nodes/usuaris. CoDiP2P contempla un sistema de *buffers* on tracta els peers workers que volen adherir-se a l' execució de l' aplicació. Aquests buffers tenen una limitació pel que fa al nombre de peers que poden encabir.

3. L' elecció dels nombres primers escollits. L' opció escollida dels nombres primers emprats ha estat la dels nombres primers forts. Un nombre primer $p$ fort és aquell que compleix que $p = a \cdot q + 1$ on $a$ és un enter $\lll p$ i $q$, un nombre primer gran. Un nombre primer segur computacionalment gran és, segurament, un primer fort criptogràficament. Aquest aspecte augmenta la seguretat i, per tant, dificulta el criptoanàlisi.

## 7.2 Conclusions i futures línies de treball

En aquest treball final de carrera, s' ha implementat l' algorisme Rho de Pollard sobre el grup multiplicatiu $\mathbb{F}_p^*$ en la seva versió paral·lelitzada sobre la plataforma de computació distribuida CoDiP2P amb el llenguatge de programació interpretat Java.
Amb aquest projecte, s' ha volgut posar en èmfasi la capacitat o potencial del sistema CoDiP2P en front un dels problemes matemàtics més coneguts i més estudiats en relació a la importància dels àmbits en què s' utilitza.
Les conclusions d' aquest treball final de carrera són positives, no tant pels resultats obtinguts, sino pels coneixements adquirits. Hem pogut conèixer un dels problemes més importants de la criptografia moderna, quines debilitats té i quins possibles atacs s' estudien per al seu criptoanàlisi. Així també, hem conegut la plataforma CoDiP2P, desenvolupada a la Universitat de Lleida, sobre la que hem executat l' algorisme Rho de Pollard. Aquest era un dels principals objectius d' aquest treball. L' execució de l' algorisme Rho de Pollard paral·lelitzat sobre un clúster ja va correspondre a l' objectiu de treballs de final de carrera passats com [15] o [16].
Fins al moment, la plataforma CoDiP2P comptava amb només 2 problemes matemàtics amb els quals poder comprovar la seva estabilitat i, per tant, eficiència: la suma dels primers $n$ nombres naturals i la cerca de nombres primers de Mersenne. Amb el problema del logaritme discret, s' ha posat a prova el seu sistema *Task Dispatcher*, o despatxador de tasques, on han intervingut fins a 19 nodes. Si bé, el que hom en primer terme se li ocorre per tal de millorar els resultats obtinguts és una ampliació del hardware disponible per tal d' augmentar el potencial de càlcul, la conclusió principal és que l' estudi de millors algorismes, o variants d' aquests algorismes, poden paliar les deficiències que pugui tenir el hardware de què es disposa.
Com a futures línies de treball podem plantejar-nos les següents fites que aconsegueixin aproximar-nos a un atac al logaritme discret emprant l' algorisme Rho de Pollard més efectiu:

- Un estudi acurat del ple funcionament de la plataforma CoDiP2P, analitzant-ne el codi font i observant les seves limitacions més fonamentals.
- Un estudi més en profunditat del mètode presentat per M.J. Wiener i P.C. van Oorschot per a la paral·lelització de la Rho de Pollard.
- Una anàlisi acurada de l' algorisme exposat en [15] basat en una comunicació molt més directa entre els peers de la xarxa i una actualització freqüent dels seus estats.

# Bibliografia i Netgrafia

[1] Alfred J. Menezes, Paul C. Van Oorschot and Scott A. Vanstone. Handbook of Applied Crypyography, October 1996.

[2] Fabian Kuhn, René Struik. Random Walks Revisited: Extensions of Pollard's Rho Algorithm for Computing Multiple Discrete Logarithms. Springer-Verlag London, UK. 2001.

[3] Faith E. Fich. Lower Bounds for the Cycle Detection Problem. In Proceedings of STOC, pp.96-105. 1981.

[4] Ignasi Barri Vilardell i Josep Maria Rius Torrentó. CoDiP2P: Sistema de Computación Distribuida Peer to Peer. Escola Politècnica Superior, Universitat de Lleida, Lleida. Marzo de 2008.

[5] Joan Gimbert, Xavier Hernández, Nacho López, Josep M. Miret, Ramiro Moreno i Magda Valls. Curs pràctic d' àlgebra per a informàtics. Departament de Matemàtica, Escola Politècnica Superior, Universitat de Lleida, Lleida. 2004.

[6] John M. Pollard. A Monte Carlo method for factorization. BIT, v. 15, Nº 3. 1975.

[7] John M. Pollard. Monte Carlo Methods for Index Computation (mod p). Mathematics Department, Berkshire, England. July 1978.

[8] Paul C. Van Oorschot and Michael J. Wiener. Parallel Collision Search with Cryptanalytic Applications. Ontario, Canada. September 1996.

[9] Richard P. Brent. An improved Monte Carlo factorization algorithm. Department of Computer Science, Australian National University, Canberra, Australia, BIT 20, 1980.

[10] R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, Vol. 21 (2), pp.120-126. 1978.

[11] Shi Bai and Richard P. Brent. On the Efficiency of Pollard's Rho Method for Discrete Logarithms. Department of Computes Science, Australian National University, Canberra, Australia. 2008.

[12] Simon Singh. Los códigos secretos; El arte y la ciencia de la criptografía desde el antiguo Egipto a la era Internet. Trad. de josé Ignacio Moraza. Barcelona: Editorial Debate, 2000.

[13] Steven D. Galbraith and Raminder S. Riprai. An Improvement to the

Gaudry-Schost Algorithm for Multidimensional Discrete Logarithm Problems. Heidelberg, Germany: Springer-Verlag Berlin, 2009.

[14] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Vol. IT-31, Nº 4, July 1985.

[15] Teodoro Andrés Laírla Morlans. Paralelización del algoritmo Rho de Pollard utilizando el protocolo de paso por mensajes MPI. Escola Politècnica Superior, Universitat de Lleida, Lleida. Septiembre de 2010.

[16] Tomàs Bigordà Soldevila. Atac al problema del logaritme discret mitjançant la Rho de Pollard. Escola Politècnica Superior, Universitat de Lleida, Lleida. Setembre, 2005.

[17] Whitfield Diffie; Martin E. Hellman. New directions in Cryptography. IEEE Transactions on Information Theory, Vol.22 (6), pp.644-654. November, 1976.

[18] http://ca.wikipedia.org/wiki/Atac_Man-in-the-middle

[19] http://en.wikipedia.org/wiki/Birthday_problem